Past-paper walk-through

Methods: Passing and Returning References of an Object ▪ 3.6

eXercise

Questions in this set

- 2021 Q2
- 2014 Q4

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 2

2021 ▪ Q2

The class SingleTable represents a table at a restaurant.

```
k+kdpublic+w k+kdclass n+ncSingleTable
p
+w    c+cm/** Returns the number of seats at this table. The value is always greater than or equal to
+w    k+kdpublic+w k+ktint+w n+nfgetNumSeatsp(p)
+w    p+w   c+cm/* implementation not shown */+w   p

+w    c+cm/** Returns the height of this table in centimeters. */
+w    k+kdpublic+w k+ktint+w n+nfgetHeightp(p)
+w    p+w   c+cm/* implementation not shown */+w   p
+w    c+cm/** Returns the quality of the view from this table. */
+w    k+kdpublic+w k+ktdouble+w n+nfgetViewQualityp(p)
+w    p+w   c+cm/* implementation not shown */+w   p

+w    c+cm/** Sets the quality of the view from this table to value. */
+w    k+kdpublic+w k+ktvoid+w n+nfsetViewQualityp(k+ktdouble+w nvaluep)
+w    p+w   c+cm/* implementation not shown */+w   p

+w    c+c1// There may be instance variables, constructors, and methods that are not shown.
```

Question 2

At the restaurant, customers can sit at tables that are composed of two single tables pushed together. You will write a class `CombinedTable` to represent the result of combining two `SingleTable` objects, based on the following rules and the examples in the chart that follows.

- A `CombinedTable` can seat a number of customers that is two fewer than the total number of seats in its two `SingleTable` objects (to account for seats lost when the tables are pushed together).
- A `CombinedTable` has a desirability that depends on the views and heights of the two single tables. If the two single tables of a `CombinedTable` object are the same height, the desirability of the `CombinedTable` object is the average of the view qualities of the two single tables.
- If the two single tables of a `CombinedTable` object are not the same height, the desirability of the `CombinedTable` object is 10 units less than the average of the view qualities of the two single tables.

Question 2

Assume `SingleTable` objects `t1`, `t2`, and `t3` have been created as follows.

- `SingleTable t1` has 4 seats, a view quality of 60.0, and a height of 74 centimeters.
- `SingleTable t2` has 8 seats, a view quality of 70.0, and a height of 74 centimeters.
- `SingleTable t3` has 12 seats, a view quality of 75.0, and a height of 76 centimeters.

The chart contains a sample code execution sequence and the corresponding results.

Statement	Value Returned (blank if no value)	Class Specification
`CombinedTable c1 = new CombinedTable(t1, t2);`		A `CombinedTable` is composed of two `SingleTable` objects.
`c1.canSeat(9);`	true	Since its two single tables have a total of 12 seats, c1 can seat 10 or fewer people.
`c1.canSeat(11);`	false	c1 cannot seat 11 people.
`c1.getDesirability();`	65.0	Because c1's two single tables are the same height, its desirability is the average of 60.0 and 70.0.
`CombinedTable c2 = new CombinedTable(t2, t3);`		A `CombinedTable` is composed of two `SingleTable` objects.
`c2.canSeat(18);`	true	Since its two single tables have a total of 20 seats, c2 can seat 18 or fewer people.
`c2.getDesirability();`	62.5	Because c2's two single tables are not the same height, its desirability is 10 units less than the average of 70.0 and 75.0.
`t2.setViewQuality(80);`		Changing the view quality of one of the tables that makes up c2 changes the desirability of c2, as illustrated in the next line of the chart. Since `setViewQuality` is a `SingleTable` method, you do not need to write it.
`c2.getDesirability();`	67.5	Because the view quality of t2 changed, the desirability of c2 has also

Question 2

The last line of the chart illustrates that when the characteristics of a `SingleTable` change, so do those of the `CombinedTable` that contains it.
Write the complete `CombinedTable` class. Your implementation must meet all specifications and conform to the examples shown in the preceding chart.

Solution 2

- Model solution (class CombinedTable, 9 pts):
- “‘java
- public class CombinedTable
- {
- private SingleTable table1;
- private SingleTable table2;
- public CombinedTable(SingleTable tab1, SingleTable tab2)
- {

Solution 2

- table1 = tab1;
- table2 = tab2;
- }
- public boolean canSeat(int n)
- {
- if (table1.getNumSeats() + table2.getNumSeats() - 2 >= n)
- {
- return true;
- }

Solution 2

- else
- {
- return false;
- }
- }
- public double getDesirability()
- {
- if (table1.getHeight() == table2.getHeight())
- {

Solution 2

- return (table1.getViewQuality() + table2.getViewQuality()) / 2;
- }
- else
- {
- return ((table1.getViewQuality() + table2.getViewQuality()) / 2) - 10;
- }
- }
- }
- ‘‘‘

Solution 2

- Points (1 each, 9 total):
- 1. Declares class header ‘class CombinedTable‘ and constructor header ‘CombinedTable(SingleTable ____, SingleTable ____)‘ (must not be private).
- 2. Declares appropriate private instance variables including at least two SingleTable references.
- 3. Constructor initializes the instance variables using its parameters.
- 4. Declares header ‘public boolean canSeat(int ____)‘.
- 5. Calls ‘getNumSeats‘ on a SingleTable object.
- 6. ‘canSeat(n)‘ returns true if and only if the sum of seats of the two tables minus 2 is $>=$ n.

Solution 2

- 7. Declares header ‘public double getDesirability()‘.
- 8. Calls ‘getHeight‘ and ‘getViewQuality‘ on SingleTable objects.
- 9. ‘getDesirability‘ computes and returns the average of the two tables’ view qualities when heights are equal, or that average minus 10 when heights differ (return is not assessed separately from the correct calculation, but the value must actually be returned).

Question 4

2014 ▪ Q4

The menu at a lunch counter includes a variety of sandwiches, salads, and drinks. The menu also allows a customer to create a "trio," which consists of three menu items: a sandwich, a salad, and a drink. The price of the trio is the sum of the two highest-priced menu items in the trio; one item with the lowest price is free. Each menu item has a name and a price. The four types of menu items are represented by the four classes `Sandwich`, `Salad`, `Drink`, and `Trio`. All four classes implement the following `MenuItem` interface.

```
k+kdpublic+w k+kdinterface n+ncMenuItem
p
+w   c+cm/** @return the name of the menu item */
+w   nString+w n+nfgetNamep(p)p;

+w   c+cm/** @return the price of the menu item */
+w   k+ktdouble+w n+nfgetPricep(p)p;
p
```

Question 4

The following diagram shows the relationship between the `MenuItem` interface and the `Sandwich`, `Salad`, `Drink`, and `Trio` classes.
[Diagram: a UML-style relationship diagram. At the top, a box labeled "interface MenuItem". Below it, an arrow points up from four boxes to the interface box, indicating each implements MenuItem: "Sandwich", "Salad", "Drink", "Trio".]
For example, assume that the menu includes the following items. The objects listed under each heading are instances of the class indicated by the heading.

Sandwich	**Salad**	**Drink**
"Cheeseburger" 2.75	"Spinach Salad" 1.25	"Orange Soda" 1.25
"Club Sandwich" 2.75	"Coleslaw" 1.25	"Cappuccino" 3.50

Question 4

Question 4 continues on page 13

The menu allows customers to create `Trio` menu items, each of which includes a sandwich, a salad, and a drink. The name of the `Trio` consists of the names of the sandwich, salad, and drink, in that order, each separated by "/" and followed by a space and then "Trio". The price of the `Trio` is the sum of the two highest-priced items in the `Trio`; one item with the lowest price is free.

Question 4

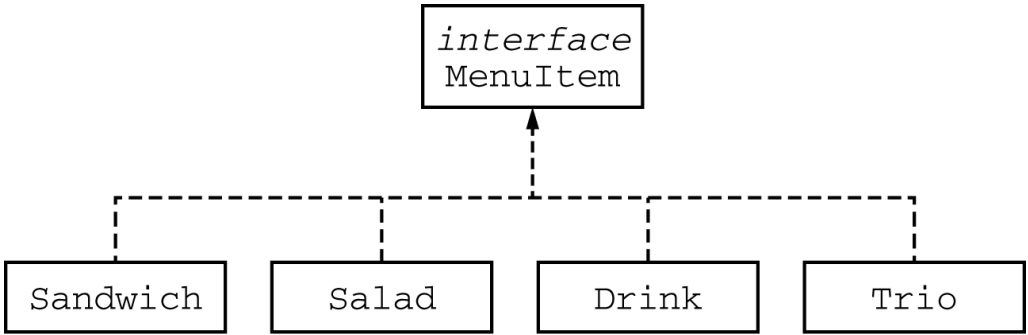
A trio consisting of a cheeseburger, spinach salad, and an orange soda would have the name "Cheeseburger/Spinach Salad/Orange Soda Trio" and a price of $4.00 (the two highest prices are $2.75 and $3.50).
Write the `Trio` class that implements the `MenuItem` interface. Your implementation must include a constructor that takes three parameters representing a sandwich, a salad, and a drink. The following code segment should have the indicated behavior.

```
nSandwich+w nsandwichp;
nSalad+w nsaladp;
nDrink+w ndrinkp;
c+cm/* Code that initializes sandwich, salad, and drink */

nTrio+w ntrio+w o=+w knew+w nTriop(nsandwichp,+w nsaladp,+w ndrinkp)p;+w  c+c1// Compiles without err

nTrio+w ntrio1+w o=+w knew+w nTriop(nsaladp,+w nsandwichp,+w ndrinkp)p;+w c+c1// Compiletime error
nTrio+w ntrio2+w o=+w knew+w nTriop(nsandwichp,+w nsaladp,+w nsaladp)p;+w c+c1// Compiletime error
```

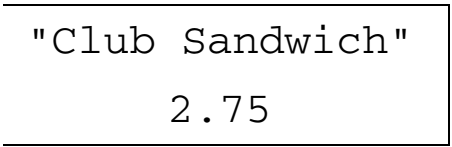
Question 4

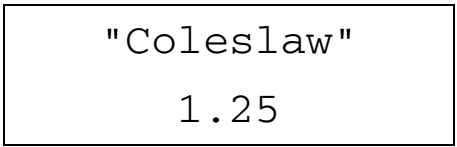
WRITE YOUR SOLUTION ON THE NEXT PAGE.
Write the complete `Trio` class below.

Question 4



Question 4





Question 4